Subject: Understanding packages, assemblies, and nests
Posted by eldiener on Fri, 13 Mar 2015 04:24:22 GMT
View Forum Message <> Reply to Message

I read the doc on packages, assemblies, and nests but I don't understand the difference between an assembly and a nest. Both are explained in terms of being a collection of packages, but I could not make out how they are related to packages in a different manner.

Most IDEs have projects, which can build an assortment of exes/libraries etc., each with a different configuration. This would be the equivalent of an assembly building packages it seems. So I don't understand what "nests" are and how they fit in the hierarchy.

The IDE is confusing to say the least. Isn't there some easy way to just open a project which tells the IDE what to build ?

Subject: Re: Understanding packages, assemblies, and nests
Posted by dolik.rce on Fri, 13 Mar 2015 05:28:42 GMT
View Forum Message <> Reply to Message

eldiener wrote on Fri, 13 March 2015 05:24I read the doc on packages, assemblies, and nests but I don't understand the difference between an assembly and a nest. Both are explained in terms of being a collection of packages, but I could not make out how they are related to packages in a different manner.
Hi eldiener,

Nest and assemblies are two levels of the same thing. Nest is a collection of somehow related packages (e.g. MyApps are my personal packages, bazaar are comunity contributed packages etc.) and is represented on hard drive as a single directory. Assembly is actually a collection of nests. That allows you to simply organize the packages in flexible way, without having multiple copies of the same package or all the packages mixed together.

You can use this for example to test newer versions of U++: Just configure two assemblies, one with nests MyApps + uppsrc-stable (pointing to some older stable version, that you already tested) and second with MyApps + uppsrc-fresh (pointing to latest sources, which you'd like to upgrade to). Now you can build the same package from MyApps with two different U++ versions in parallel, simply by opening it from differen assembly.

TheIDE might have some rough edges, but IMHO the package management is actually one of it's strongest points

Best regards,
Honza

Subject: Re: Understanding packages, assemblies, and nests

Posted by eldiener on Fri, 13 Mar 2015 06:20:57 GMT
View Forum Message <> Reply to Message

dolik.rce wrote on Fri, 13 March 2015 01:28eldiener wrote on Fri, 13 March 2015 05:24I read the doc on packages, assemblies, and nests but I don't understand the difference between an assembly and a nest. Both are explained in terms of being a collection of packages, but I could not make out how they are related to packages in a different manner.
Hi eldiener,

Nest and assemblies are two levels of the same thing. Nest is a collection of somehow related packages (e.g. MyApps are my personal packages, bazaar are comunity contributed packages etc.) and is represented on hard drive as a single directory. Assembly is actually a collection of nests. That allows you to simply organize the packages in flexible way, without having multiple copies of the same package or all the packages mixed together.

You can use this for example to test newer versions of U++: Just configure two assemblies, one with nests MyApps + uppsrc-stable (pointing to some older stable version, that you already tested) and second with MyApps + uppsrc-fresh (pointing to latest sources, which you'd like to upgrade to). Now you can build the same package from MyApps with two different U++ versions in parallel, simply by opening it from differen assembly.

TheIDE might have some rough edges, but IMHO the package management is actually one of it's strongest points

Best regards,
Honza

When I open the IDE its "Select main package" lists assemblies and for each assembly there is a list of packages. But from your explanation above, where a nest is a collection of packages and an assembly is a collection of nests, I would expect that selecting a main package would involve selecting an assembly, which would list its nests, and then each nest would list its packages. This is what confuses me.

Subject: Re: Understanding packages, assemblies, and nests
Posted by dolik.rce on Fri, 13 Mar 2015 06:39:15 GMT
View Forum Message <> Reply to Message

eldiener wrote on Fri, 13 March 2015 07:20When I open the IDE its "Select main package" lists assemblies and for each assembly there is a list of packages. But from your explanation above, where a nest is a collection of packages and an assembly is a collection of nests, I would expect that selecting a main package would involve selecting an assembly, which would list its nests, and then each nest would list its packages. This is what confuses me.

By default it shows only packages from the first nest in the assembly with main configuration (meaining the can be build standalone), as those are the ones most likely to be opened. You can change it to display other packages as well using the select box in lower left corner.

Honza

Subject: Re: Understanding packages, assemblies, and nests
Posted by eldiener on Sat, 14 Mar 2015 00:41:41 GMT
View Forum Message <> Reply to Message

dolik.rce wrote on Fri, 13 March 2015 02:39eldiener wrote on Fri, 13 March 2015 07:20When I open the IDE its "Select main package" lists assemblies and for each assembly there is a list of packages. But from your explanation above, where a nest is a collection of packages and an assembly is a collection of nests, I would expect that selecting a main package would involve selecting an assembly, which would list its nests, and then each nest would list its packages. This is what confuses me.

By default it shows only packages from the first nest in the assembly with main configuration (meaining the can be build standalone), as those are the ones most likely to be opened. You can change it to display other packages as well using the select box in lower left corner.

Honza

I see this but I still do not understand the relationships between assemblies, nests, packages, main packages, sub-packages etc. What do these mean in terms of building applications and libraries with their dependencies ?

BTW, please don't take this personally but the documentation on assemblies, nests, and packages is absolutely horrendous. Doesn't anybody know how to explain programming ideas among the Ultimate++ developers ? Perhaps from my many questions the Ultimate++ developers will understand why these docs in this area are so bad. I am not spamming, just trying to understand the package management system.

So let me put down what i have gleaned from the docs and maybe by having my many questions answered I can use the Ultimate++ IDE in some sort of manner that will let me create cross-platform apps and libraries. First I will ask about packages, since that is at the lowest level between assemblies, nests, and packages.

1) Is a package a single build of an application/library etc. ?
2) Why is something a main package and something else is not a main package ? Is it because a main package can have dependencies but is never a dependent of another package ?
3) For any given package how can I see its hierarchy of dependencies ?
4) Given a package already created somewhere in a directory, how do I add it to the IDE so it knows about it ?

Subject: Re: Understanding packages, assemblies, and nests

Sorry about confusion.

First of all, 'package' is perhaps the wrong name of entity. A couple of years ago I was suggesting renaming to 'unit' or 'module', but the name is already too stuck...

Quote:
1) Is a package a single build of an application/library etc. ?

I guess shortest description is 'minimal library module in source form'.

E.g. good example for starters is 'plugin/gif' - that is package that contains code dealing with 'gif' format.

Of course, some packages contain more stuff, like 'Core', which contains all U++ basics.

Quote:
2) Why is something a main package and something else is not a main package ? Is it because a main package can have dependencies but is never a dependent of another package ?

Main package represents (in most cases) application. It is 'main', because it is the one package you choose when starting theide. Rest of packages are put into the project based on dependencies.

Quote:
3) For any given package how can I see its hierarchy of dependencies ?

In 'Package organizer'.

Quote:
4) Given a package already created somewhere in a directory, how do I add it to the IDE so it knows about it ?

Package is a directory with some source files and ".upp" file, which contains description of package (list of files, dependecies ('uses'), less often optionally external libraries, compiler options etc...).

Nest is directory which contains (at first level) packages.

Assembly is an ordered set of nests (so that all packages of all nests are combined into single 'namespace').

Now I am not sure how your question is related to these concepts, but let us say you have a

directory of source files and you want to 'convert' that to package.

I guess simple way to do that is to create a fresh empty package, copy source files to package folder, then add them into package using right click on list of package files and choosing 'Add package directory files...'.

Subject: Re: Understanding packages, assemblies, and nests
Posted by mirek on Sat, 14 Mar 2015 06:43:49 GMT
View Forum Message <> Reply to Message

eldiener wrote on Fri, 13 March 2015 07:20I would expect that selecting a main package would involve selecting an assembly, which would list its nests, and then each nest would list its packages. This is what confuses me.

Well, originally when you selected 'assembly' (when starting theide), you got list of all main packages of all nests (because assembly is supposed to mix them into single namespace).

Anyway, for practical reasons, the dialog got selector, which by default only lists main packages of the first nest of assembly.

The reason for that is that the first nest usually contains main packages you want to work on when you have selected the assembly.

E.g. after installation, you will have 'MyApps' assembly, which combines 'MyApps' nest directory supposed to contain your projects and 'uppsrc' nest directory that contains U++ package (in essence, U++ library). When you start theide, you perhaps will want to choose your project to work on, so there is no reason to mix e.g. 'ide' main package into the list (but you can do that by changing selector).

Subject: Re: Understanding packages, assemblies, and nests
Posted by eldiener on Sat, 14 Mar 2015 17:21:33 GMT
View Forum Message <> Reply to Message

mirek wrote on Sat, 14 March 2015 02:36Sorry about confusion.

First of all, 'package' is perhaps the wrong name of entity. A couple of years ago I was suggesting renaming to 'unit' or 'module', but the name is already too stuck...

Quote:
1) Is a package a single build of an application/library etc. ?

I guess shortest description is 'minimal library module in source form'.

E.g. good example for starters is 'plugin/gif' - that is package that contains code dealing with 'gif' format.

Of course, some packages contain more stuff, like 'Core', which contains all U++ basics.

Quote:
2) Why is something a main package and something else is not a main package ? Is it because a main package can have dependencies but is never a dependent of another package ?

Main package represents (in most cases) application. It is 'main', because it is the one package you choose when starting theide. Rest of packages are put into the project based on dependencies.

Quote:
3) For any given package how can I see its hierarchy of dependencies ?

In 'Package organizer'.

Quote:
4) Given a package already created somewhere in a directory, how do I add it to the IDE so it knows about it ?

Package is a directory with some source files and ".upp" file, which contains description of package (list of files, dependecies ('uses'), less often optionally external libraries, compiler options etc...).

Nest is directory which contains (at first level) packages.

Assembly is an ordered set of nests (so that all packages of all nests are combined into single 'namespace').

Now I am not sure how your question is related to these concepts, but let us say you have a directory of source files and you want to 'convert' that to package.

I guess simple way to do that is to create a fresh empty package, copy source files to package folder, then add them into package using right click on list of package files and choosing 'Add package directory files...'.

From your answers and other explanations here are the things I assume. Please correct me if my understanding of any of this is wrong:

IDE builds a package, which generates some final file usually an executable or library but it may be any resultant file. The package can contain dependencies in the form of other packages. So when a package is built it may be necessary to build any of its dependencies first before

rebuilding the main package.

Regarding the "main package": are not all packages represented by a directory with a .upp file ? if that is so how does the IDE know what is a main package as opposed to what is a dependent package ?

I see the Package organizer. A visual tree-like display of dependencies might be nice.

Regarding the IDE and adding a package to it I think you missed my point. Suppose I already have a package somewhere on my file system with its .upp file. How do I tell the IDE that the package exists and it should be listed somewhere by the IDE ?

Finally if I have a package does its dependencies have to be within the same directory structure of the main package ? I hope that is not the case.

Subject: Re: Understanding packages, assemblies, and nests
Posted by dolik.rce on Sat, 14 Mar 2015 19:18:53 GMT
View Forum Message <> Reply to Message

eldiener wrote on Sat, 14 March 2015 18:21IDE builds a package, which generates some final file usually an executable or library but it may be any resultant file. The package can contain dependencies in the form of other packages. So when a package is built it may be necessary to build any of its dependencies first before rebuilding the main package.Yes, that is correct. I'm not aware about any case where the result would be something else than library (either static or dynamic) or executable, but in theory other result formats could be possible as well.

eldiener wrote on Sat, 14 March 2015 18:21Regarding the "main package": are not all packages represented by a directory with a .upp file ? if that is so how does the IDE know what is a main package as opposed to what is a dependent package ?In TheIDE there is a "Main package configuration" dialog (in Project menu, or when you click on the select box in menubar), which lists the build flags that can be used for the package. Any package that has one or more entries in this list is considered to be "main" (because non-main packages would't require build flags).

eldiener wrote on Sat, 14 March 2015 18:21I see the Package organizer. A visual tree-like display of dependencies might be nice.It is not simple tree, multiple packages can share same dependencies. For example, have a look at the graph in this thread if you're interested.

eldiener wrote on Sat, 14 March 2015 18:21Regarding the IDE and adding a package to it I think you missed my point. Suppose I already have a package somewhere on my file system with its .upp file. How do I tell the IDE that the package exists and it should be listed somewhere by the IDE ?You just set the assembly to contain the nest where your package is located - that is it's parent directory. If you put this nest at first position in the assembly, then the package (given it is "main") will be listed when you start TheIDE and select this assembly.

eldiener wrote on Sat, 14 March 2015 18:21Finally if I have a package does its dependencies have to be within the same directory structure of the main package ? I hope that is not the case.No, the dependencies can be in any of the nests in the same assembly. Usually, your

package would by in MyApps, and it would depend on for example on Core, which is located in the uppsrc nest.

Honza

Subject: Re: Understanding packages, assemblies, and nests
Posted by eldiener on Sun, 15 Mar 2015 03:20:34 GMT
View Forum Message <> Reply to Message

dolik.rce wrote on Sat, 14 March 2015 15:18eldiener wrote on Sat, 14 March 2015 18:21IDE builds a package, which generates some final file usually an executable or library but it may be any resultant file. The package can contain dependencies in the form of other packages. So when a package is built it may be necessary to build any of its dependencies first before rebuilding the main package.Yes, that is correct. I'm not aware about any case where the result would be something else than library (either static or dynamic) or executable, but in theory other result formats could be possible as well.

eldiener wrote on Sat, 14 March 2015 18:21Regarding the "main package": are not all packages represented by a directory with a .upp file ? if that is so how does the IDE know what is a main package as opposed to what is a dependent package ?In TheIDE there is a "Main package configuration" dialog (in Project menu, or when you click on the select box in menubar), which lists the build flags that can be used for the package. Any package that has one or more entries in this list is considered to be "main" (because non-main packages would't require build flags).

eldiener wrote on Sat, 14 March 2015 18:21I see the Package organizer. A visual tree-like display of dependencies might be nice.It is not simple tree, multiple packages can share same dependencies. For example, have a look at the graph in this thread if you're interested.

eldiener wrote on Sat, 14 March 2015 18:21Regarding the IDE and adding a package to it I think you missed my point. Suppose I already have a package somewhere on my file system with its .upp file. How do I tell the IDE that the package exists and it should be listed somewhere by the IDE ?You just set the assembly to contain the nest where your package is located - that is it's parent directory. If you put this nest at first position in the assembly, then the package (given it is "main") will be listed when you start TheIDE and select this assembly.

eldiener wrote on Sat, 14 March 2015 18:21Finally if I have a package does its dependencies have to be within the same directory structure of the main package ? I hope that is not the case.No, the dependencies can be in any of the nests in the same assembly. Usually, your package would by in MyApps, and it would depend on for example on Core, which is located in the uppsrc nest.

Honza

I do not understand why non-main packages do not have build flags. Don't non-main packages have to be built also ? Suppose I create a shared library use the IDE. I assume a shared library is another package, just like anything else being built in the IDE from source files. Don't I need build flags to tell the IDE how to build the shared library ?

Regarding have the IDE add a package which already exists as a .upp file: are you saying that this package needs to go in an already existing assembly ? Or can I create a new assembly for the package ? Are assemblies just a grouping mechanism for packages in the IDE so that a user of the IDE can open the package by finding it within a particular assembly ? Can a package be in more than one assembly, ie. can different assemblies have the same nest in its list of nests ? I also see that one can choose a non-main package to be opened in the IDE, but the menu item says 'File | Set main package...'. Isn't this a misnomer since the end-user can choose a non-main package to build ?

You wrote "Usually, your package would by in MyApps, and it would depend on for example on Core, which is located in the uppsrc nest." Does this mean that packages which the end-user creates should go into the MyApps assembly ? I am gathering that the other assemblies are for Ultimate++'s own use, except maybe for MyAppsWBazaar.

My understanding of assemblies and nests are (please correct me if I am wrong):

1) Assemblies are grouping mechanisms for finding a package in the IDE. Every assembly has a list of nests.
2) A nest is just a directory where packages may be found in that directory or one of its subdirectories.
3) Neither an assembly or a nest exists when building something in the IDE, only a package gets built with possibly its package dependencies.
3) A package is what gets built within the IDE. A package may have other packages dependencies. A main package is never a dependent of any other package.

Subject: Re: Understanding packages, assemblies, and nests
Posted by mirek on Sun, 15 Mar 2015 07:15:15 GMT
View Forum Message <> Reply to Message

Quote:
I do not understand why non-main packages do not have build flags. Don't non-main packages have to be built also ? Suppose I create a shared library use the IDE. I assume a shared library is another package, just like anything else being built in the IDE from source files. Don't I need build flags to tell the IDE how to build the shared library ?

This is not the purpose of build flags. They are usually used as 'binary-wide' settings. Some of them are recognized by build system (e.g. MT says that application is multithreaded, which affects the way how packages are compiled, GUI says that it is gui application, which is recognized by Win32 builders etc...), some are application specific, e.g. developer can decide to recognize 'DEMO' build flag, which would restrict the functionality of resulting executable. Your main package can have multiple main configurations.

Quote:
Regarding have the IDE add a package which already exists as a .upp file: are you saying that this package needs to go in an already existing assembly ? Or can I create a new assembly for

the package ? Are assemblies just a grouping mechanism for packages in the IDE so that a user of the IDE can open the package by finding it within a particular assembly ?

Yes. It is like include path or say system 'PATH' (for finding executables).

Quote:
Can a package be in more than one assembly, ie. can different assemblies have the same nest in its list of nests ?

Definitely!

More importantly, you can use assemblies to e.g. select version of library. Say you have 'stable' libraries and 'experimental' - you can have two assemblies, one will group your main project with experimental, other with stable library.

Quote:
I also see that one can choose a non-main package to be opened in the IDE, but the menu item says 'File | Set main package...'. Isn't this a misnomer since the end-user can choose a non-main package to build ?

True. Practical needs sometimes blur things... perhaps it would be more correct to have 'main package' and 'package with main configuration' as two distinct terms, but later is too long...

Quote:
You wrote "Usually, your package would by in MyApps, and it would depend on for example on Core, which is located in the uppsrc nest." Does this mean that packages which the end-user creates should go into the MyApps assembly ? I am gathering that the other assemblies are for Ultimate++'s own use, except maybe for MyAppsWBazaar.

It is something like initial suggested setup. E.g. when I start theide on my work laptop, I have about 40 assemblies in the list

Quote:
3) A main package is never a dependent of any other package.

Actually, there is no such limitation. Main package in fact can be used in another main package dependency. There is even a macro 'flagMAIN' that is only defined when package is build as main.

Usage scenario: Imagine I would want to create text editor which would provide much of functionality that 'theide' provides, but I would be too lazy to correctly split ide main package sources into correct 'non-main' package. I could have all sources of theide included into new main package by simply adding theide into now project and perhaps putting some of #ifdefs flagMAIN

into it. It is not typical, perhaps not even recommended, but sometimes ends justify means...

(rest of your points is correct).

Mirek

Subject: Re: Understanding packages, assemblies, and nests
Posted by eldiener on Sun, 15 Mar 2015 16:02:08 GMT
View Forum Message <> Reply to Message

mirek wrote on Sun, 15 March 2015 03:15Quote:
I do not understand why non-main packages do not have build flags. Don't non-main packages have to be built also ? Suppose I create a shared library use the IDE. I assume a shared library is another package, just like anything else being built in the IDE from source files. Don't I need build flags to tell the IDE how to build the shared library ?

This is not the purpose of build flags. They are usually used as 'binary-wide' settings. Some of them are recognized by build system (e.g. MT says that application is multithreaded, which affects the way how packages are compiled, GUI says that it is gui application, which is recognized by Win32 builders etc...), some are application specific, e.g. developer can decide to recognize 'DEMO' build flag, which would restrict the functionality of resulting executable. Your main package can have multiple main configurations.

Quote:
Regarding have the IDE add a package which already exists as a .upp file: are you saying that this package needs to go in an already existing assembly ? Or can I create a new assembly for the package ? Are assemblies just a grouping mechanism for packages in the IDE so that a user of the IDE can open the package by finding it within a particular assembly ?

Yes. It is like include path or say system 'PATH' (for finding executables).

Quote:
Can a package be in more than one assembly, ie. can different assemblies have the same nest in its list of nests ?

Definitely!

More importantly, you can use assemblies to e.g. select version of library. Say you have 'stable' libraries and 'experimental' - you can have two assemblies, one will group your main project with experimental, other with stable library.

Quote:
I also see that one can choose a non-main package to be opened in the IDE, but the menu item says 'File | Set main package...'. Isn't this a misnomer since the end-user can choose a non-main

package to build ?

True. Practical needs sometimes blur things... perhaps it would be more correct to have 'main package' and 'package with main configuration' as two distinct terms, but later is too long...

Quote:
You wrote "Usually, your package would by in MyApps, and it would depend on for example on Core, which is located in the uppsrc nest." Does this mean that packages which the end-user creates should go into the MyApps assembly ? I am gathering that the other assemblies are for Ultimate++'s own use, except maybe for MyAppsWBazaar.

It is something like initial suggested setup. E.g. when I start theide on my work laptop, I have about 40 assemblies in the list

Quote:
3) A main package is never a dependent of any other package.

Actually, there is no such limitation. Main package in fact can be used in another main package dependency. There is even a macro 'flagMAIN' that is only defined when package is build as main.

Usage scenario: Imagine I would want to create text editor which would provide much of functionality that 'theide' provides, but I would be too lazy to correctly split ide main package sources into correct 'non-main' package. I could have all sources of theide included into new main package by simply adding theide into now project and perhaps putting some of #ifdefs flagMAIN into it. It is not typical, perhaps not even recommended, but sometimes ends justify means...

(rest of your points is correct).

Mirek

Thanks for your explanations !

I still do not understand the difference between a main package and a non-main package as far as Ultimate++ is concerned. Why divide your idea of a package into these two distinct categories ? In other words what does a main package consist of that a non-main package does not have and why is this important to either the IDE or to a person using Ultimate++ to create executables or libraries to be run cross-platform ?

Your explanation suggests, although I might be misunderstanding it, that a main package can have more than one way of building it but a non-main package only is built in one particular way. Is this the distinction between them ? Because if it is I can conceive of very few things which are built in only one way. A library as well as an executable often has different build configurations. But if this is the single distinction between a main package and a non-main package then I do understand the difference. I just don't understand why this distinction is important to the IDE or the

programmer using the Ultimate++ IDE.

Also I think I do understand assemblies/nests as merely grouping mechanisms in the IDE in order to specify a package or a particular build of a package in a particular category (assembly). If they exist as anything else please tell me.

Subject: Re: Understanding packages, assemblies, and nests
Posted by mirek on Sun, 15 Mar 2015 16:21:37 GMT
View Forum Message <> Reply to Message

Quote:
I still do not understand the difference between a main package and a non-main package as far as Ultimate++ is concerned. Why divide your idea of a package into these two distinct categories ? In other words what does a main package consist of that a non-main package does not have and why is this important to either the IDE or to a person using Ultimate++ to create executables or libraries to be run cross-platform ?

Well, in fact, the main reason is that when starting theide, you have to choose the project what you want to work with. There is a lot of packages, but at the moment of starting theide, usually only those with main configuration are those you eventually choose from (because they represent applications you are developing). That is all the magic and reason. Other than that, main and non-main are essentially the same.

Quote:
Also I think I do understand assemblies/nests as merely grouping mechanisms in the IDE in order to specify a package or a particular build of a package in a particular category (assembly). If they exist as anything else please tell me.

Sure. You have to specify where package directories are, right?

Mirek

Subject: Re: Understanding packages, assemblies, and nests
Posted by eldiener on Sun, 15 Mar 2015 18:16:26 GMT
View Forum Message <> Reply to Message

mirek wrote on Sun, 15 March 2015 12:21Quote:
I still do not understand the difference between a main package and a non-main package as far as Ultimate++ is concerned. Why divide your idea of a package into these two distinct categories ? In other words what does a main package consist of that a non-main package does not have and why is this important to either the IDE or to a person using Ultimate++ to create executables or libraries to be run cross-platform ?

Well, in fact, the main reason is that when starting theide, you have to choose the project what you want to work with. There is a lot of packages, but at the moment of starting theide, usually only those with main configuration are those you eventually choose from (because they represent applications you are developing). That is all the magic and reason. Other than that, main and non-main are essentially the same.

Quote:
Also I think I do understand assemblies/nests as merely grouping mechanisms in the IDE in order to specify a package or a particular build of a package in a particular category (assembly). If they exist as anything else please tell me.

Sure. You have to specify where package directories are, right?

Mirek

OK. I get it as far as main packages and non-main packages. Actually a project for me is usually the development of libraries and not applications. I assume that Ultimate++ components can be incorporated in a library as well as in an application.

Thanks for your help ! I only wish your documentation on packages, assemblies, and nests had explained these basics as well as you have done here in this thread. If I have further questions about using the IDE I will start a new thread.
